

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/310518500>

# Towards Generating Realistic SNMP-MIB Dataset for Network Anomaly Detection

Article · September 2016

CITATIONS

8

READS

551

3 authors:



**Mouhammd Alkasassbeh**

Princess Sumaya University for Technology

54 PUBLICATIONS 234 CITATIONS

[SEE PROFILE](#)



**Ghazi Al-Naymat**

Princess Sumaya University for Technology

43 PUBLICATIONS 285 CITATIONS

[SEE PROFILE](#)



**Eshraq Hawari**

Mu'tah University

7 PUBLICATIONS 12 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Big Data Platforms Benchmark [View project](#)



abnormality in computer networks [View project](#)

# Towards Generating Realistic SNMP-MIB Dataset for Network Anomaly Detection

Mouhammd Al-kasassbeh, Ghazi Al-Naymat, Eshraq Al-Hawari

## Abstract

The enormous growth in computer networks and in Internet usage in recent years, combined with the growth in the amount of data exchanged over networks, have shown an exponential increase in the amount of malicious and mysterious threats to computer networks. Among many security issues, network attack is a major one. For example, Denial of Service (DoS) flooding attacks have recently become attractive to attackers, and these have posed devastating threats to network services. Therefore, the intrusion detection and network anomalies become very critical tasks in the field of network security research area. Researchers suffer from the lack of real-life datasets. Most of the datasets in hand depend on simulated-based approaches, which cannot represent the exact and the nature of network intrusion and anomaly scenarios. Hence, generating realistic datasets is very important as it allows for accurate and appropriate evaluation of the detection techniques. To overcome such shortcoming of the existing datasets, in this paper, we identify the important requirements to generate effective dataset and we also identify important attack scenarios and the method of injecting them in such data. Our systematic approach involves the investigation of Simple Network Management Protocol (SNMP) for network anomaly detection. For that, we present a Management Information Base (MIB) based mechanism capturing realistic SNMP-MIB statistical data. Then we use this data from an SNMP agent by means of real-life experiments involving six types of DoS attacks and Brute Force attack. Our dataset consists of 4998 records, where each record consists of 34 MIB variables, which are categorized into their corresponding groups, namely: Interface, IP, TCP and ICMP.

**Keywords:** *Anomaly detection, Attacks, DoS, SNMP, MIB.*

## 1 Introduction

Due to the growing confidence on the Internet and its wide-ranging connectivity, Internet attacks can cause impending damage to the systems being used. A well-known protection system used to monitor the network is called Intrusion Detection System (IDS), which plays a critical role in detecting types of attack that occur in a network and provides some mechanisms to repair the caused damage, or reduce the possibility of future harm. Having said that, an important component for designing an effective IDS solution is to have an excellent IDS evaluation dataset. This research suffers from a major problem that is; the lack of real datasets. In this paper,

we aim to illustrate the process of preparing real dataset to help researchers to examine their proposed techniques and produce valid results to the community. The below sections introduce important terminologies such as, the network anomaly, types of attacks, and network protocols.

## **1.1 Network Anomaly**

Network anomaly is a deviation in the normal behavior of a network in the presence of any intruder in a network or due to network overload. These anomalous events disrupt the normal functionality of network services. Normal network behavior can be characterized by various factors, such as the type of network data to be measured, traffic volume of the network and the type of applications that are running on the network. Moreover, Intrusion Detection System (IDS) is the process of monitoring any abnormal activity occurring in a computer system or network and comparing it with a normal event to identify signs of intrusion. Intrusion refers to a malicious activity aimed at compromising the confidentiality, integrity and availability of network components in an attempt to disrupt the security policy of networks [1]. IDS major functionality is not only for monitoring, it issues alerts about the abnormal activities and, optionally, to limit such activities [2]. In the past decade, IDS has been recognized as an intense research area due to the rapid increase in sophisticated attacks on networks [3]. By implementing the IDS concept, network monitoring and network security can be conducted more precisely.

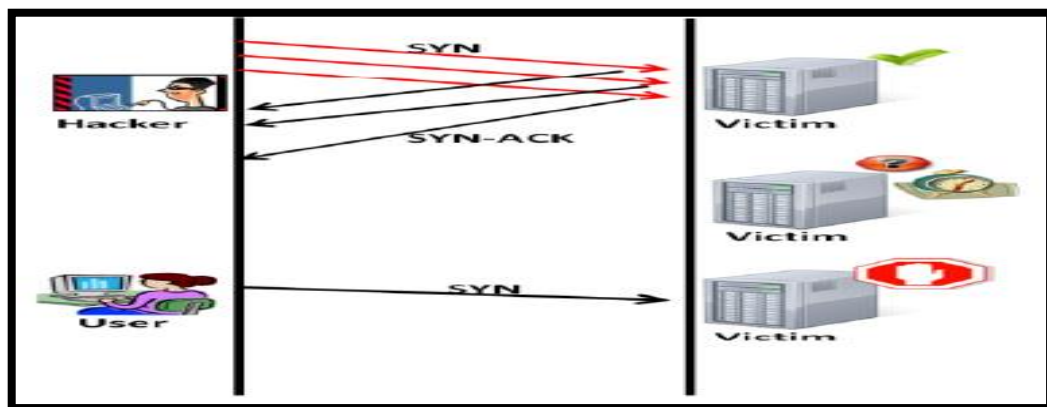
## **1.2 Network Attacks**

A network attack is any process or method that is used to carry out malicious attempts against computer networks to compromise network security [4]. Attackers consider various distinct stages when they want to carry out attacks, starting from the initial motivation of the attackers to the final execution of the attacks [5]. The most attractive types of attack to the attackers are Denial of Service (DoS) attacks. The focus of our work is on the detection of attacks; specifically DoS flooding and Brute Force attacks.

DoS attack is one of the network attacks that aims to deprive legitimate users of the services offered by a server on the Internet or the target network. This can be done by flooding a server (victim) with a high volume of traffic to consume all the server's resources (CPU, memory and bandwidth) and to stop it from processing the user's legitimate requests. As a result, this creates network congestion on the target system, thus

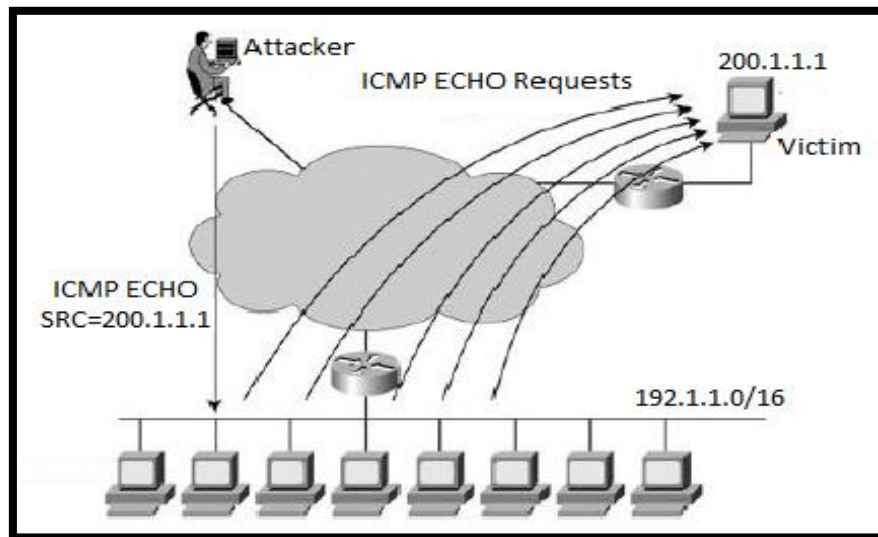
disrupting normal operations and the victim's services become unavailable [5], [6]. Typically, DoS attacks can be launched from either a local or remote location and be implemented diversely with a variety of tools and codes. Furthermore, DoS attack is considered one of the most dangerous threats to the security of computer networks. Its effects can range from a minor increase in the service response time to a complete unavailability, while also having serious financial implications for organizations. A recent report by Amazon shows that even a 100ms delay in the service response time causes an approximate 1% drop in their overall sales [5]. Another version of the DoS attack is Distributed Denial of Service (DDoS) attack. It sends huge amount of traffic to the victim by compromising insecure machines in the network causing denial of service [7]. Following are examples on common Types of DoS Flooding Attack;

1. **TCP-SYN Attack:** The TCP SYN flood attack exploits a vulnerability of the three-way handshake [Synchronize (SYN), SYN-ACK (Acknowledgment), and ACK] mechanism that is used between the host and the server to establish a connection. During the attack, the attacker sends SYN packets with source IP (Internet Protocol) addresses (spoofed) that do not exist to the server. The server responds to a SYN request by sending SYN ACK, stores the request information in the memory stack and waits for ACK (confirmation) from the client. While the server is waiting for ACK from the client, the request remains in the memory stack. The server will not receive ACK packets because the source IP address has spoofed the IP address. If the attacker sends multiple SYN requests within very short intervals, the requests will fill up the entire memory stack and then the server's resources will be exhausted and it becomes unable to respond to further requests from the legitimate user. Figure 1 depicts the method of a TCP-SYN flood attack [8].



**Figure 1:** TCP-SYN Attack.

2. **UDP Flood Attack:** This type of attack is caused by sending or flooding UDP packets towards random ports of the server. The server opens the packet and finding nothing in it, then sends the unreachable packet back to the destination. Because of this huge amount of UDP packets, the victim's resources (bandwidth) will be exhausted and can be busy serving these requests. This leads to unavailability for the legitimate users. A UDP flood attack can be more effective in smaller networks as the size of the UDP packets are enormous [9].
3. **ICMP-ECHO Attack:** This attack focuses on flooding the victim's bandwidth. According to the ICMP protocol, when a device on the network receives a "ping" request, it replies to the source IP address with a message informing of the status of the receiver device. During this attack, the attacker crafts a large number of ICMP packets with a spoofed source IP address the same as that of the victim's IP address and sends them through ICMP echo requests to a broadcast address. All devices on the network that have received these requests will reply with messages back to the victim. Eventually, all the reply messages will exhaust the victim's resources. The strength of an ICMP-ECHO attack depends on the number of devices that receive the requests and the attack packet rates [10]. Figure 2 illustrates the ICMP attack method.

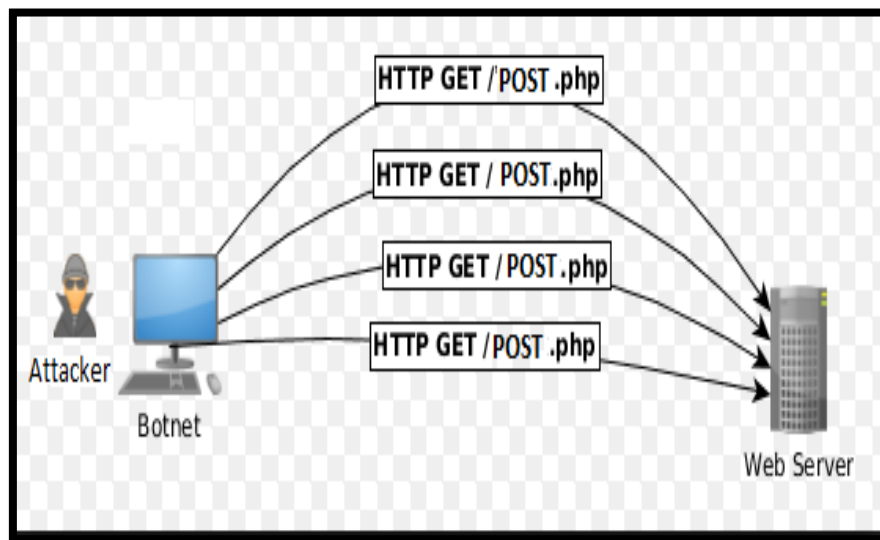


**Figure 2:** ICMP-ECHO Attack [8].

4. **HTTP Flood Attack:** This attack is also called an HTTP GET/POST flooding attack. It is also considered a non-spoofed attack. In this attack, the attacker aims to attack web servers and applications to consume a huge amount of the victim's resources. Attackers send a large number of valid HTTP requests (get/post) to a victim, as shown

in Figure 3. Such requests are often sent by botnets where each of the bots can generate a large number of valid requests (usually more than 10 requests a second). Here, session connection request rates from the attackers are higher than the session connection requests from the normal users.

An HTTP flooding attack may be one of the most non-vulnerability threats that web servers may encounter because it is hard to distinguish between malicious HTTP traffic and normal HTTP traffic [2].



**Figure 3:** HTTP (GET/POST) Attack.

5. Slowloris Attack: This attack is also called a Slow Header attack. During the attack, the attacker with a non-spoofed IP address sends sessions with high workload requests. These requests are partial HTTP header requests that are slow to update, rapidly and continuously grow and never close. The attack continues until all available sockets are taken up by the requests, and the web server becomes unavailable to any legitimate connection. The slowloris attack can cause a web server to crash, using a limited number of machines or even a single machine without any side effects on other services and ports [2].
6. Slowpost Attack: This attack is also called a Slow Request Bodies and first appeared in 2010. This attack is similar to a Slowloris attack in that the attackers send sessions with high workload requests to bring down web servers. In this attack, the attacker sends a complete HTTP header request, which defines the content length field in the post message body as this request is sent for normal traffic. The data

are then sent to fill the message body at a rate one byte every two minutes, and at the same time the server remains waiting for each message body to be completed, leading to denial of web services [2].

### **1.3 Simple Network Management Protocol**

The SNMP is a popular protocol for managing network devices in an Internet using the TCP/IP protocol suite. SNMP is an application layer protocol and runs over the User Datagram Protocol (UDP). It is used for configuring, and collecting information from network devices, such as Normal PCs, switches, servers and routers [11]. Since its creation in 1980s, SNMP has been the solution for managing network devices in various network types with widespread acceptance because of its simplicity [12]. Most of the current research in intrusion detection depends on analyzing raw packet data to evaluate the security status of computer systems and networks, this will lead to a significant processing burden and late detection time [13]. Another data source about network traffic provided by several network management protocols such as CIMP (Common Management Information Protocol), RMON (Remote Network Monitoring), and SNMP. The most deployed and widely used protocol is the SNMP [14].

As we mentioned above, SNMP support variables that correspond to traffic information at the device level. This information from the network devices can be passively monitored and could be used to characterize network behavior and, therefore, can be used for network anomaly detection [11].

A Management Information Base (MIB) is a database designed to handle the objects in a system network. MIB is often accompanying with SNMP. Since SNMP [13] has been standardized and already implemented in all network devices, add to the availability of the MIB statistical data to easily collect for analysis, thus we have selected the SNMP as our protocol for network attacks detection. By utilizing the fine-grained data provided from MIB in anomaly detection, some of the challenges in network intrusion detection can be avoided.

For every malicious activity that happens, a specific MIB variable will be affected in some way. Hence SNMP-MIB data are a source of indicators for anomaly detection. For more accurate network anomaly detection, proper SNMP-MIB variables must be selected because there is no single variable that is capable of capturing all anomalies on networks, so that the number of MIB variables involved is minimized and the range of anomalies covered is maximized [13].

## 1.4 Contribution and Paper Organization

In this paper, we make the following contributions: (1) we illustrate the configuration and design of a real-life test-bed for generating attack traffic. (2) We describe the SNMP-MIB statistical data collected from our designed test-bed. (3) A total of 4998 records each consists of 34 MIB variables are made available for researchers to test their IDS solutions.

The paper is organized as follows: Section 2 describes the recent work that used SNMP-MIB simulated-based data. The dataset generation approach is described thoroughly in Section 3 followed by our discussion in Section 4.

## 2 Background

Many studies have exploited SNMP-MIB data in the early detection of network anomalies. Various methodologies and techniques have been proposed and evaluated. Some researchers have presented approaches based on statistical analysis of MIB data, while others have recently utilized Machine Learning techniques to detect network attacks and other anomalies. We will review previous related works on anomaly detection using SNMP-MIB. The first attempt to exploit SNMP for network security is reported in Cabrera et al. [15]. They proposed a methodology for the early detection of Distributed Denial of Service (DDoS) attacks by applying statistical tests for causality to extract MIB variables that contain precursors to attacks. The proposed methodology depends on using 91 MIB traffic variables from 5 groups (IP, ICMP, TCP, UDP and SNMP) collected periodically from the target and the attacker participating in attacks. Three types of DDoS attack (Ping Flood, Targa3 and UDP Flood) were conducted with controlled loads in traffic. Their work has shown that it is possible to extract a precursor to a DDoS attack using MIB traffic variables and to detect these attacks before the target is shut down. However, in our work, we used unlike and extra types of DDoS attack and different MIB variables, as well as, tried to investigate and injected up-to-date attacks that help researchers for testing their techniques for detection the abnormality of the traffic.

Ramah et al. [16] developed an anomaly detection system implemented in Matlab for anomalies detection based on network traffic collected periodically from SNMP-MIB data that included four MIB variables from an interface group. They performed two DoS attacks (SMURF, SYN-Flood attack) and a scan attack to evaluate the developed system in a real



experiment and they found that this system could detect flooding attacks efficiently. Unfortunately, four MIB variables are not enough to give a generic solution to detect all the DoS attacks, as in a network traffic classification, what is abnormal to a network is normal to others, it depends on the type of the traffic that they use.

Park & Kim [17] proposed a fast and lightweight algorithm for intrusion detection; especially for traffic flooding attacks detection, based on the correlation of SNMP-MIB data. They selected 16 MIB-II variables from 6 groups (system, IF, IP, ICMP, TCP, and UDP) to be used for attack detection. They performed 3 types of flooding attack (TCP-SYN, UDP and ICMP flooding) to test the performance of the proposed detection algorithm in a real experiment. They only selected specific MIB variables, for example, in an interface group the selected two variables as same as the TCP group. However, in our case, we investigate all the variables in each group. Our point was to give the researchers the chance to study all the MIB variables as each variable and group is affected by different attacks. It should be known that there is no single variable that can catch all the abnormal behaviors in the networks also there is no master MIB.

Hoque et al. [18] adopted a statistical method depending on the Kolmogorov Smirnov test (KST) for anomaly detection on computer networks using SNMP-MIB data. They periodically polled 22 MIB variables corresponding to 5 MIB groups (Interface, TCP, IP, ICMP, UDP) from 24 interfaces of one switch during their experiment. The proposed algorithm was tested against several types of DoS flooding attack, including TCP SYN flooding, UDP flooding, ICMP PING flooding and IP flooding. Though, the interface group has most of the MIB variables, the other groups have few for example from the UDP group they used only two variables. As we mentioned earlier, some of the variables have weight more than others for specific attacks.

Namvarasl & Ahmadzadeh [19] have also presented an intrusion detection system based on SNMP-MIB and machine learning approaches. Their system consisted of three modules: the first for selecting key MIB variables from three classification algorithms (C4.5, RIPPER and attribute selection), and the second for generating an intrusion detection model based on the chosen variables and detecting DoS/DDoS attacks in real time in the third module. The dataset used in their system consisted of appropriate MIB-II variables among 66 variables corresponding to 4 MIB

groups (IP, ICMP, TCP, and UDP) and involving a TCP-SYN flood attack, UDP flood attack and ICMP flood attack. As known one of the most important group is the Interface group, which they miss in their work.

Al-Kasassbeh [20] has adopted the distributed model in order to exclude the scalability problems in the network. His work showed that the statistical methods based on the Wiener filter that upgraded to the mobile agent could be used to detect the abnormality attempts. He took advantage of the correlation matrix between the input MIB variables and the cross-correlation with the desired MIB variables to detect abnormal situations. Al-Kasassbeh used only limited number of MIB variables.

It should be known that all the aforementioned techniques depend on generating their own datasets that are tailored to their research problems. However, this research is mainly designed to generate a generic/comprehensive dataset that can be exploited in any type of research problem in the same context (IDS). In addition, we aimed in this research to choose the most important MIB variables from all groups.

### **3 Dataset Generation**

#### **3.1 Work Model**

The most important point in using SNMP-MIB data for anomaly detection is that the proper SNMP-MIB variables must be used because there is no single variable that is capable of capturing all anomalies on networks. Thus in this work, we focused on using the effective variables for more accurate anomalies detection.

We concentrated on the router device to collect the MIB variables. We selected 34 MIB variables from 5 MIB groups in MIB-II defined in [21]: Interface (variables of this group were collected from a particular interface of the router), IP, TCP, UDP, and ICMP. The data types of the 34 MIB variables were counter 32, which are a non-negative 4-byte integer that is continuously incremented from 0 to  $2^{32}$ ; when it reaches its maximum value, it wraps back to 0. By means of comprehensive investigation, we selected these variables among other MIB variables in the groups because they are affected more by the attack traffic where these variables are continuously updated with the incoming and outgoing traffic over the network; thus, they could be more effective for attack detection.

The MIB-II groups with their corresponding variables that are investigated and used are as follows:

- a) **Interface group:** This group is not related to a specific layer, it defines information about all the interfaces of the node including interface number, physical address, and IP address. We selected 8 MIB variables from this group:
  1. ifInOctets : This variables represents the total number of octets received on the interface, including framing characters.
  2. ifOutOctets: The total number of octets transmitted out of the interface, including framing characters.
  3. ifoutDiscards: The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted.
  4. ifInUcastPkts: The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were not addressed to a multicast or broadcast address at this sub-layer.
  5. ifInNUcastPkts: The number of packets, delivered by this sub-layer to a higher (sub-)layer, which were addressed to a multicast or broadcast address at this sub-layer.
  6. ifInDiscards: The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol.
  7. ifOutUcastPkts: The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer.
  8. ifOutNUcastPkts: The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent.
- b) **IP group:** This group provides network layer information related to IP, such as the routing table and the IP address. We selected 8 MIB variables from this group:
  1. ipInReceives: The total number of input datagrams received from interfaces, including those received in error.
  2. ipInDelivers: The total number of input datagrams successfully delivered to IPv4 user-protocols (including ICMP).
  3. ipOutRequests: The total number of IPv4 datagrams which local IPv4 user protocols (including ICMP) supplied to IPv4 in requests for transmission.

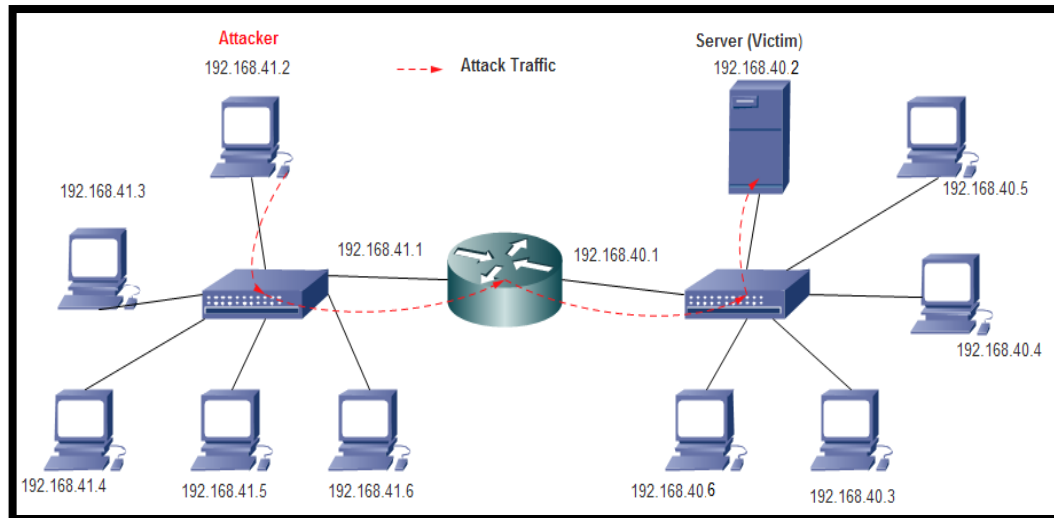
4. **ipOutDiscards**: The number of output IPv4 datagrams for which no problem was encountered to prevent their transmission to their destination, but which were discarded.
  5. **ipInDiscards**: The number of input IPv4 datagrams for which no problems were encountered to prevent their continued processing, but which were discarded.
  6. **ipForwDatagrams**: The number of input datagrams for which this entity was not their final IPv4 destination, as a result of which an attempt was made to find a route to forward them to that final destination.
  7. **ipOutNoRoutes**: The number of IPv4 datagrams discarded because no route could be found to transmit them to their destination. Note that this includes any datagrams, which a host cannot route because all of its default routers are down.
  8. **ipInAddrErrors**: The number of input datagrams discarded because the IPv4 address in their IPv4 header's destination field was not a valid address to be received at this entity. this counter includes datagrams discarded because the destination address was not a local address.
- c) **ICMP group**: This group defines information related to ICMP, such as the number of packets sent and received and total errors created. We selected 6 MIB variables from this group:
1. **icmpInMsgs**: The total number of ICMP messages which the entity received.
  2. **icmpInDestUnreaches**: The number of ICMP Destination Unreachable messages received.
  3. **icmpOutMsgs**: The total number of ICMP messages which this entity attempted to send.
  4. **icmpOutDestUnreaches**: The number of ICMP Destination Unreachable messages sent.
  5. **icmpInEchos**: The number of ICMP Echo (request) messages received.
  6. **icmpOutEchoReps**: The number of ICMP Echo Reply messages sent.
- d) **TCP group**: This group provides transport layer information related to TCP, such as the connection table, time-out value, number of ports, and number of packets sent and received. We selected 8 MIB variables from this group:
1. **tcpOutRsts**: The number of TCP segments sent containing the RST flag.

2. **tcpInSegs**: The total number of segments received, including those received in error. This count includes segments received on currently established connections.
  3. **tcpOutSegs**: The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets.
  4. **tcpPassiveOpens**: The number of times TCP connections have made a direct transition to the SYN state.
  5. **tcpRetransSegs**: The total number of segments retransmitted; that is, the number of TCP segments transmitted containing one or more previously transmitted octets.
  6. **tcpCurrEstab**: The number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.
  7. **tcpEstabResets**: The number of times that TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state.
  8. **tcpActiveOpens**: The number of times that TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state.
- e) **UDP group**: This group provides transport layer information related to UDP, such as the number of ports and number of packets sent and received. We selected 4 MIB variables from this group:
1. **udpInDatagrams**: The total number of UDP datagrams delivered to UDP users.
  2. **udpOutDatagrams**: The total number of UDP datagrams sent from this entity.
  3. **udpInErrors**: The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port.
  4. **udpNoPorts**: The total number of received UDP datagrams for which there was no application at the destination port.

### 3.2 Test-bed Configuration

The scenarios were conducted in a real environment in one of the labs of Information Technology Department at Mu'tah University to collect SNMP-MIB data. We constructed a test-bed network to carry out an actual experimental attack. Figure 4 illustrates the network topology, which consisted of one router, 2 switches and 2 subnets. The first subnet contained 5 PCs where one of them acted as the attacker. The second

subnet contained 5 PCs and one of them acted as the server (victim). The specifications of the test-bed network are shown in Table 1.



**Figure 4:** Test-bed network architecture used during the DoS dataset generation

**Table 1.** Test-bed network specifications.

Machine Type	Hardware Description	Software Installed
PC	Processors: Intel (R) Core (TM) i3-2120 CPU@ 3.30GHZ, Memory: 2.00 GB, Network adapters: Realtek PCIe GBE Family Controller.	OS: windows 7. Lan Traffic v2.
Server (victim)	Processors: Intel (R) Core (TM) i3-2120 CPU@ 3.30GHZ, Memory: 2.00 GB, Network adapters: Realtek PCIe GBE Family Controller.	OS: windows 7. Lan Traffic v2. (Apache web server, VNC server and FTP server).
PC Attacker	Processors: Intel (R) Core (TM) i3-2120 CPU@ 3.30GHZ, Memory: 2.00 GB, Network adapters: Realtek PCIe GBE Family Controller.	OS: windows 7. Lan Traffic v2. Tools of attack (Hyenae FE, HttpFlood2.5.1, THC-Hydra, slowloris script and HttpDosTool4.0).
PC router	Processor: Intel (R) Celeron (R) D CPU 3.07GHZ, Memory: 2.00GB, Networkadapters: 1) Intel(R) PRO/100 VE Network Connection. 2) Realtek RTL8139/810x Family Fast Ethernet NIC.	OS: windows 7.
Switch	Cisco Catalyst 2950 .	

### 3.3 MIB Dataset Collection

In this setup, different tools were used to generate normal traffic as well as attack traffic over the test-bed network. Since the test-bed network was isolated from the Internet, traffic needed to be generated between these two subnets in order to simulate a real network so that the environment could

be as realistic as possible. For this reason the LanTrafficV2<sup>1</sup> was used to generate normal traffic over the network during the period of the experiment. This tool generates normal traffic by using TCP, UDP, ICMP or SCTP protocols, allowing the user to define data size and packet parameters as well as the delay between each packet sent; it also allows creating, editing and send-receive scenarios by using the add-on software called the Automation Tool for LanTraffic V2.

In our setup, the LanTraffic software was installed on all the PCs in the two subnets. The source and destination IP addresses were configured so that each PC could send TCP, UDP or ICMP packets to another PC and *vice versa*. Different scenarios with different parameters were written and carried out between the two subnets. Network traffic load was distributed as TCP (70%), UDP (30%) and the normal traffic was generated approximately at a rate up to 50 Mbps over the network during the experiment. On the other hand, the attack traffic generation, out of the five PCs in the first subnet, 1 PC (attacker) was used to generate attack traffic, and at the same time, the rest of the PCs were used to send and receive normal traffic. The attack traffic was sent against the server (victim machine).

### 3.4 Attacks tools

To generate the attack traffic, we used well-known open source tools, which are given below with a brief description:

1. **THC- Hydra 5.2**<sup>2</sup>: This is a very fast, versatile and well-known Brute Force password cracking tool that has been successfully used with most modern network security protocols, such as FTP, SSH and HTTP. It can perform rapid dictionary attacks against more than 50 protocols, including ftp, http, https, telnet, and much more. The THC-Hydra tool is available for different types of operating systems, such as Windows, Solaris, Linux and others.
2. **HyenaeFE**<sup>3</sup>: This is an advanced and highly flexible platform-independent network packets generator tool. It allows the user to generate several well-known attacks, such as DoS, DDoS, and other attack scenarios. The HyenaeFE tool is an interactive attack assistant and comes with a clusterable remote daemon [22].

---

<sup>1</sup> <http://www.zti-communications.com/download-free-15-day-trial-version/>

<sup>2</sup> <https://github.com/maaaaz/thc-hydra-w>

<sup>3</sup> <http://www.toolwar.com/2013/10/hyenae-tools.html>

3. **DOSHTTP<sup>4</sup> 2.5.1**: This tool is a powerful HTTP Flood DoS Testing Tool for Windows. It is an easy-to-use tool that uses multiple asynchronous sockets to launch an effective HTTP Flood; it can be used simultaneously on multiple clients to carry out DDoS attacks.
4. **Slowloris script<sup>5</sup> and ActivePerl language<sup>6</sup>**: Slowloris is a piece of software developed by Robert “RSnake” Hansen. It is effective attack software that enables a single computer to take down a web server with minimal bandwidth and without any side effects on other services and ports. Slowloris has proven to be highly effective against many popular types of web server software, such as the Apache server. Perl is an open source scripting language, which is widely used for system administration and programming on the Internet. ActivePerl is a distribution package of Perl for Windows. It comes ready to install on Windows and other systems.
5. **HttpDosTool4.0<sup>7</sup>**: This is a free-to-use tool that can execute different types of Denial of Service attacks. It started in early 2000 for testing the availability and performance of a web application towards DoS attacks.

These tools provide a number of configuration parameters, such as packet type (TCP, UDP, ICMP) number of packets sent, packet size, delay, etc. These can be very useful in generating attacks. In this paper, we conducted a Brute Force attack and six types of DoS flooding attack, carried out by using the previously mentioned tools with different time durations and parameters, as explained below. The attack types with the corresponding tools are illustrated in Table 2.

---

<sup>4</sup> <http://doshttp.soft32.com/>

<sup>5</sup> <http://ha.ckers.org/slowloris/slowloris.pl>

<sup>6</sup> <http://www.activestate.com/activeperl>

<sup>7</sup> <https://code.google.com/p/owasp-dos-http-post/>



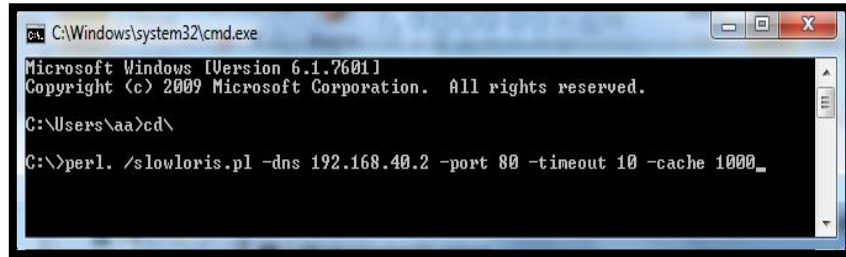
**Table 2.** Attack types and the corresponding tools used.

Category	Attack type	Tool used
<i>DoS flooding attacks</i>	TCP-SYN flooding	HyenaeFE
	UDP flooding	HyenaeFE
	ICMP-ECHO flooding	HyenaeFE
	HTTP flood	DOSHTTP 2.5.1
	Slowloris	Slowloris script
	Slow post	HttpDosTool4.0
<i>Brute Force attack</i>	Brute Force attack	THC-Hydra

As illustrated in Table 2, we executed the attacks by using different tools. Each type of attack was launched against the server (victim) in different scenarios. The TCP-SYN attack, UDP Flood and ICMP-ECHO attack were executed using the same tool (HyenaeFE packet generator tool). The 3 attacks were launched with different attack parameters (TCP packet type, packet payload, packet send duration, packet send delay) as provided by the Hyenae tool. The attack execution procedure and the parameters were determined by commands in the Hyenae tool.

- 1. TCP-SYN attack:** We carried out the TCP-SYN attack more than once in 3 scenarios. Each scenario was performed within a specific time period and with different parameters.
- 2. UDP Flood attack:** The UDP flooding attack was performed in 2 scenarios within a specific time period for each scenario.
- 3. ICMP-ECHO attack:** We launched the ICMP-ECHO attack in 3 scenarios. In each scenario, the ICMP packets were sent for a specific time period and with specific parameters.
- 4. HTTP Flood attack:** This attack was performed by the DoSHTTP tool. We performed the attack in 3 scenarios within a specific time period for each scenario. In each scenario, we continuously sent HTTP requests with different asynchronous socket numbers to perform an effective HTTP Flood attack.
- 5. Slowloris Attack:** We used Slowloris software and ActivePerl script to launch this kind of attacks. We launched it 3 times in 3 scenarios. Each time we launched the attack against the server (victim) on port 80 for a specific period with a specific number of sockets (packets). When we performed this command, the victim was flooded with

1000 sockets every 10 second timeout connection. These parameters were sufficient to cause the server to crash in 3 minutes in our experiments. Figure 5 shows a screenshot of the Slowloris attack and its execution by sending 1000 sockets every 10 seconds.

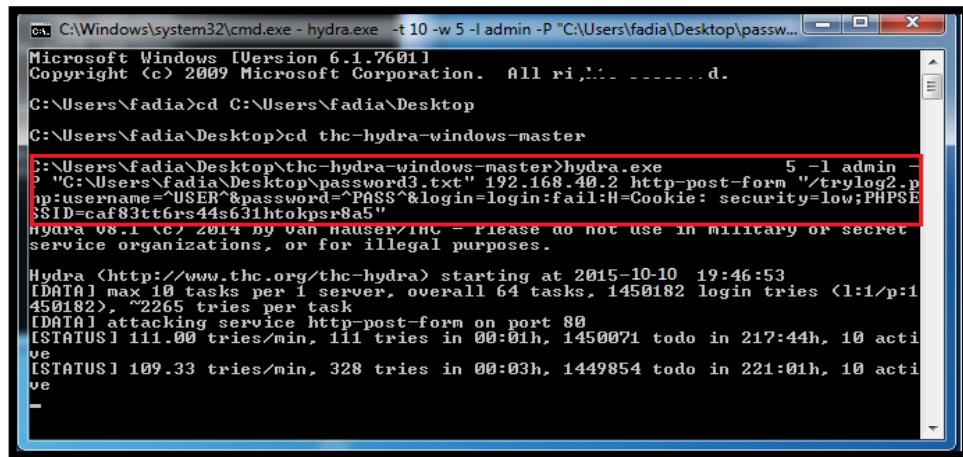


**Figure 5.** Slowloris Attack Command.

6. **Slowpost Attack:** We executed the Slowpost attack using the HttpDosTool4.0 tool in 2 scenarios. In each scenario, we sent slow HTTP requests with different parameters, including: number of **connection** requests to be sent to the victim, **connection rate** during the request, **timeout** in seconds to determine the time duration between connection requests, in addition to the **content length** field in one of the HTTP request messages.
7. **Brute Force Attack:** We used the THC-Hydra tool to perform the Brute Force attack. In our experiments, we created login website pages on Apache server and wrote commands to execute it. In the commands, we used a passwords file with about 1450183 passwords, and an admin word as the username for password cracking. The command below shows how we performed the Brute Force attack against the Apache server with the aim of cracking the password.

```
hydra.exe -l admin -P "C:\Users\fadia\Desktop\password3.txt" 192.168.40.2  
http-post-form  
"/trylog2.php:username=^USER^&password=^PASS^&login=login:fail:H=  
Cookie: security=low;PHPSESSID=caf83tt6rs44s631htokpsr8a5"
```

We conducted the attack in 2 scenarios, running the brute force continuously until the correct password was found. Figure 6 shows a screenshot of the running Brute Force attack.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\fadia>cd C:\Users\fadia\Desktop
C:\Users\fadia\Desktop>cd thc-hydra-windows-master
C:\Users\fadia\Desktop\thc-hydra-windows-master>hydra.exe -t 10 -w 5 -l admin -P "C:\Users\fadia\Desktop\password3.txt" 192.168.40.2 http-post-form "/trylog2.php:username=\"USER\"&password=\"PASS\"&login=login:fail:H=Cookie: security=low;PHPSESSID=caf83tt6rs44s631htokpsr8a5"
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2015-10-10 19:46:53
[DATA] max 10 tasks per 1 server, overall 64 tasks, 1450182 login tries (1:1/p:1450182), ~2265 tries per task
[DATA] attacking service http-post-form on port 80
[STATUS] 111.00 tries/min, 111 tries in 00:01h, 1450071 todo in 217:44h, 10 active
[STATUS] 109.33 tries/min, 328 tries in 00:03h, 1449854 todo in 221:01h, 10 active
```

**Figure 6.** Screenshot of Running Brute Force Attack.

The last stage is to collect the MIB variables, we first activated the SNMP agent in the router to be ready to receive SNMP commands and run them on the local MIB, then send back the reply. Secondly, we wrote a JAVA program using the classes of the webNMS Agent Toolkit Java Edition 6<sup>8</sup>. The Java program uses SNMP query messages, such as GET and GET-NEXT, to read the MIB variables from the SNMP agent.

We collected MIB variables at periodic intervals (every 15 seconds) in both the normal mode and the attack mode of the network. We determined 15s for polling data from the router according to many studies, for example [11], where the authors conducted different experiments to determine the best time interval for polling MIB variables from a target device. They found that the best time interval was 15s in terms of CPU utilization and the network load during polling the data, or for fast attack detection with a low burden in the network [13]. Therefore, we decided to follow the same time interval for polling MIB variables as 15 seconds.

The 34 MIB variables were collected every 15 seconds during the duration of the attack experiments, where the server (victim) was placed under different kinds of attack, as mentioned previously. We collected a total of 4998 MIB data records as described in Table 3.

<sup>8</sup> <https://www.webnms.com/>

**Table 3.** Traffic type and number of generated records.

	Type of traffic	Number of records
1.	Normal	600
2.	TCP-SYN	960
3.	UDP flood	773
4.	ICMP-ECHO	632
5.	HTTP flood	573
6.	Slowloris	780
7.	Slowpost	480
8.	Brute Force	200

Table 4 lists basic summary statistics of the generated data (the 5-Number summary and the standard deviation). This table provides the researchers and those who are going to benefit from the generated data with some important information that highlights the nature of the used MIB variables.

**Table 4.** The 5-Number summary and the standard deviation.

Attribute Names	Min	Max	STD	Q1	Q2	Q3
ifInOctets	1426588	4294415640	1233851794	1093687078	2180395334	3221331795
ifOutOctets	161843	4294061114	1153395187	372442651	893567819. 5	2449010026
ifOutDiscards	0	196630	74228.9779	0	0	4875
ifInUcastPkts	701369	243982848	58006834.5	50197228	79062666.5	131393482
ifInNUcastPkts	2735	35238	7936.895813	15079	20987	26931
ifInDiscards	0	196630	74228.98972	0	0	4875
ifOutUcastPkts	223076	93698308	23284377.61	8048524	17450864	40394694
ifOutNUcastPkts	796	8311	1784.778092	3799	5089	6388
tcpOutRsts	1	4	1.086868566	1	2	4
tcpInSegs	55	1945	570.2247108	195	817	1286
tcpOutSegs	55	1535	457.8215205	189	654	1022
tcpPassiveOpens	3	142	42.64760765	12	59	93
tcpRetransSegs	0	24	5.583369177	0	2	14

tcpCurrEstab	0	4	0.341317667	0	0	2
tcpEstabResets	2	8	1.661978023	4	6	8
tcpActiveOpens	3	41	10.98180851	3	6	10
udpInDatagrams	8231	444247	108133.3558	117072	204983	301182
udpOutDatagrams	1316	357331	91407.98075	76502	150160	231628
udpInErrors	0	33	9.309002951	0	1	12
udpNoPorts	4	35	7.0972841	16	24	28
ipInReceives	992859	337776280	80143079.08	59957761	98121558	177484138
ipInDelivers	8811	451540	109531.1809	121595	209531	307217
ipOutRequests	1398	358471	91614.87344	76877	150836	232475
ipOutDiscards	1	1287	286.8110791	7	153	569
ipInDiscards	9	58	12.20887708	16	37	39
ipForwDatagrams	987470	337695828	80134386.46	59908837	98063382.5	177431453
ipOutNoRoutes	0	7	3.030578922	0	5	7
ipInAddrErrors	0	62	24.95423982	6	42	61
icmpInMsgs	0	127	37.0362575	12	55	84
icmpInDestUnreachs	0	68	18.8637809	11	29	44
icmpOutMsgs	0	85	20.82641141	13	34	49
icmpOutDestUnreachs	0	23	9.237201681	1	12	21
icmpInEchos	0	64	18.48188834	1	26	38
icmpOutEchoReps	0	64	18.48188834	1	26	38

To determine which of the previous attributes in the training feature vectors is most useful for discriminating between the classes to be learned. We calculated the Information Gain that tells us how important a given attribute of the feature vectors is. This will be used to decide the ordering of attributes in classification methods used.

Table 5 provides the rank of each attribute used in the data. We ranked the attributes using the Information Gain ratio for every feature so that this can show us which ones are necessary and which ones are not.

Table 5. Ranks of all attributes using information gain measure.

#	Ranked	Attribute Name	#	Rank	Attribute Name
1.	0.6338	ipOutDiscards	18.	0.4249	tcpRetransSegs
2.	0.517	icmpOutDestUnreachs	19.	0.4242	ifInUcastPkts
3.	0.501	ipInDiscards	20.	0.4238	tcpOutRsts
4.	0.4978	ifInDiscards	21.	0.4028	icmpInEchos
5.	0.4978	ifOutDiscards	22.	0.4028	icmpOutEchoReps
6.	0.4967	icmpOutMsgs	23.	0.3832	icmpInMsgs
7.	0.4898	udpNoPorts	24.	0.3732	ifOutNUcastPkts
8.	0.4804	udpInErrors	25.	0.3713	icmpInDestUnreachs
9.	0.4795	ifOutUcastPkts	26.	0.3617	ipOutNoRoutes
10.	0.4647	ipInAddrErrors	27.	0.3529	ifOutOctets
11.	0.4629	tcpEstabResets	28.	0.3473	ifInNUcastPkts
12.	0.4512	tcpInSegs	29.	0.305	udpInDatagrams
13.	0.447	tcpOutSegs	30.	0.3008	ipInDelivers
14.	0.437	tcpPassiveOpens	31.	0.2916	udpOutDatagrams
15.	0.4289	ipForwDatagrams	32.	0.2915	ipOutRequests
16.	0.4287	ipInReceives	33.	0.0736	tcpCurrEstab
17.	0.4282	tcpActiveOpens	34.	0	ifInOctets

## 4 Discussion

IDS plays a critical role in detecting types of attack that harm the computer networks and in providing some methods to repair the caused destruction, or reduce the possibility of future damage. An important component for designing an effective IDS solution is to have an excellent IDS evaluation dataset. One of the major attacks is the DoS flooding attack, which has recently attracted the attackers due to its devastating threats to network devices. Researchers started earlier in finding solutions to such problem by creating their own datasets, however, these datasets depend on simulated-based approaches. It should be known that such approaches do not represent the real nature of the network anomalies. Therefore, generating realistic datasets is very important as it allows for accurate and appropriate evaluation of the detection techniques. This paper, described the entire method to overcome the limitations of the existing datasets. We, firstly, identified the important requirements to generate effective dataset and we also illustrated the important attack scenarios and the method of injecting them in the generated data. To perform that, we investigated the SNMP for network anomaly detection and we exploited the MIB based

mechanism capturing realistic SNMP-MIB dataset. Real-life experiments we conducted to collect the SNMP-MIB statistical data involving six types of DoS attacks and Brute Force attack. The generated dataset consists of 4998 records, where each record consists of 34 MIB variables, which were experimentally chosen among many other variables. The variables data was collected every 15s and categorized into their corresponding groups, namely: Interface, IP, TCP and ICMP.

Our data generation approach allows for the ability to prove the power and effectiveness of SNMP-MIB data in network anomaly detection by demonstrating the detection of the largest possible number of the most common and modern attacks that can occur on different network layers (Network layer, Transport layer and Application layer).

Currently, we are conducting experiments to test the SNMP-MIB data using classification algorithms. In the first approach we categorized the MIB variables in 5 MIB groups (Interface, IP, ICMP, TCP and UDP) where each group included a number of MIB variables that were affiliated to. The classification algorithms were then applied to each MIB group separately, in order to show how each group is affected by attacks, and therefore to determine the most effective group(s) in anomaly detection. From the initial results of this approach, we found that the performance of each classifier is different over all the MIB groups, where the accuracy rate varied between high and low for the used classifiers. The scope of this paper is only for illustrating the data generation method.

It should be known that, we generated unbiased real-life dataset for intrusion detection purpose. The data exhibited no unintended property in both normal and the anomalous traffic for all possible network scenarios. This will allow researchers to start testing their techniques using this realistic data. To obtain the collected data you must contact the authors.

## 5 References

- [1] R. D. B. a. Chaki and Rituparna, "State of the art analysis of network traffic anomaly detection," in *Applications and Innovations in Mobile Computing (AIMoC), 2014*, IEEE, 2014, pp. 186--192.
- [2] Zargar, S. T. a. Joshi, J. a. Tipper and David, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *Communications Surveys & Tutorials, IEEE*, vol. 15, pp. 2046--2069, 2013.
- [3] M. E. H. a. T. M. Alkharobi, "Adaptive Hybrid Model for Network Intrusion Detection and Comparison among Machine Learning Algorithms," *International Journal of Machine Learning and Computing*, vol. 5, p. 17, February 2015.
- [4] Garber and Lee, "Denial-of-service attacks rip the Internet," *Computer*, pp. 12--17, 2000.
- [5] Rao, Sridhar and Subramani, "Denial of Service attacks and mitigation techniques: Real time implementation with detailed analysis," *SANS institute Infosec Reading Room Sep*, vol. 11, 2011.
- [6] J. Harju, Interviewee, [Interview]. 2011.
- [7] Mirkovic, J. a. Reiher and Peter, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, pp. 39--53, 2004.
- [8] Alqahtani, A. H. a. Iftikhar and Mohsin, "TCP/IP Attacks, Defenses and Security Tools," *International Journal of Science and Modern Engineering (IJISME)*, vol. 1, no. 10, 2013.
- [9] Salunke, M. a. Kabra, R. a. Kumar and Ashish, "Layered architecture for DoS attack detection system by combine approach of Naive bayes and Improved K-means Clustering Algorithm," vol. 2, no. 3, pp. 372-377, 2015.
- [10] K. TRESEANGRAT, "PERFORMANCE ANALYSIS OF DEFENSE MECHANISMS AGAINST UDP FLOOD ATTACKS," 2014.
- [11] Thottan, M. a. Ji and Chuanyi, "Anomaly Detection in IP Networks," *Signal Processing, IEEE Transactions on*, vol. 51, pp. 2191--2204, AUGUST 2003.
- [12] V. Bhosale, M. Sawant and F. Girkar, "Co-operative Wireless Intrusion Detection System," *International Journal of Network Security & Its Applications (IJNSA)*, vol. 4, 2012.



- [13] J. Yu, H. Lee, M.-S. Kim and D. Park, "Traffic flooding attack detection with SNMP MIB using SVMq," vol. 31, pp. 4212-4219, 2008.
- [14] Q. Wu and Z. Shao, "Network Anomaly Detection Using Time Series Analysis," in *Proceedings of the Joint International Conference on Autonomic and Autonomous Systems*, 2005.
- [15] J. B. D. Cabrera, L. Lewis, X. Qin, W. Lee and R. K. Mehra, "Proactive intrusion detection and distributed denial of service attacks—a case study in security management," *Journal of Network and Systems Management*, vol. 10, pp. 225-254, June 2002.
- [16] K. RAMAH, H. AYARI and F. KAMOUN, "Traffic Anomaly Detection and Characterization in the Tunisian National University Network," *Springer Berlin Heidelberg*, pp. 136-147, 2006.
- [17] J.-S. Park and M.-S. Kim, "Design and Implementation of an SNMP-Based Traffic," *Springer Berlin Heidelberg*, pp. 380-389, 2008.
- [18] Hoque, M. A. a. Chakraborty and Barnali, "Anomaly Based Intrusion Detection Systems," *International Journal of Computer Science Engineering and Technology*, vol. 5, no. 3, pp. 44-47, March 2015.
- [19] S. NAMVARASL and M. AHMADZADEH, "A Dynamic Flooding Attack Detection System Based on Different," *namvarasldynamic*, vol. 2, p. 279–284, SEPTEMBER 2014.
- [20] M. A. Mouhammd Al-Kasassbeh, "Network fault detection with Wiener filter-based agent," *Journal of Network and Computer Applications*, vol. 32, no. 4, pp. 824-833, 2009.
- [21] M. Rose, "Management Information Base for Network Management," 1991. [Online].
- [22] A. Alagiya, H. Joshi and A. Jani, "Performance Analysis and Enhancement of UTM Device in Local Area Network," *International Journal of Modern Education and Computer Science (IJMECS)*, vol. 5, p. 43, 2013.